HLA TESTBED DECLARATION MANAGEMENT EXPERIMENTS

Jeff Olszewski
Larry Mellon
Science Applications International Corp.
jolszewski@std.saic.com,
lmellon@std.saic.com

Richard Briggs
Virtual Technologies Corp.
rbriggs@virtc.com

KEYWORDS
HLA, Filter Space, Filters, RTI, Simulation, FCS

ABSTRACT

This paper examines the performance and usability of filter spaces, the proposed interface for HLA Declaration Management. The purpose of using filter spaces is to reduce the amount of unnecessarily reflected data between simulations in an HLA federation. Filter spaces allow simulations to request only that data which is relevant to their operation. The experiment conducted studies the performance characteristics of the RTI when the number of entities per host and number of hosts are varied. It is hypothesized that using filter spaces will significantly increase the scalability of a federation. The primary performance metrics used to evaluate scalability are packet flow between hosts and the number of reflected attribute values at remote simulations. The tests were performed in the HLA Testbed at the Integration & Evaluation Center of the Army's Topographic Engineering Center (TEC) using a militarily realistic scenario. The experiments were conducted using a two-dimensional geographic filter space, where entities subscribed to other entities and events which were within a maximum of two times their sensor range. Results of the experiment are presented and discussed.

1.0 INTRODUCTION

An important problem in distributed simulation is the large volume of unwanted network traffic which is received by simulation hosts. In a typical DIS simulation, host traffic is broadcast onto the network, requiring all hosts to receive the updates for all entities in the simulation. It is likely that many of the hosts in an exercise have no requirement to receive updates for all simulated entities. For example, they may only require updates from a particular region of the battlespace, or only require updates about a particular class of entity. Thus a potential exists for a large reduction in traffic by allowing entities to only receive data for which they express an interest.

In the DMSO High Level Architecture, the proposed mechanism for performing this interest expression is called *Declaration Management*[1]. Declaration Management uses a filtering process to prevent unwanted network traffic from reaching a host; this filtering mechanism is currently implemented in the RTI v. 0.33e as *filter spaces*. When a federation wishes to use filter spaces, it is required to define an n-dimensional space, with each dimension corresponding to an attribute which will be used to filter out unwanted data. Federates subscribe to the regions of the filter space which are of interest to them, and the RTI insures that they only receive data that updates that region of the filter space.

It is important to note that because they provide traffic reduction, filtering schemes allow the scale of an exercise to increase. If the filtering mechanisms used do not place a large burden on the host computer, system resource gains from filtering can permit the number of simulated entities to increase. This topic is discussed further in Mellon[5], which provides an overview of filtering techniques and scalability issues as they relate to the HLA.

The complete Declaration Management API can be found in [1], while filter spaces are described in more detail by Van Hook[2]. Also, a discussion of grid-based relevance filtering can be found in Van Hook [3].

This paper describes a series of HLA Declaration Management experiments which were conducted to evaluate the use of filter spaces in the RTI. These experiments were conducted at the HLA Testbed at the Integration and Evaluation Center (IEC) in Fort Belvoir in Virginia. The purpose of these experiments was not intended to yield precise results, but rather to get a sense of the trends present in the use of filter space and to get an indication of how well the filter space approach will scale in large simulations. The test process has been automated such that experiments are repeatable and follow-up experiments can be conducted.

[1] In version 1.0 of the HLA Interface Specification, this is called Data Distribution Management.

2.0 ANALYSIS

2.1 Objectives

The objectives of the declaration management experiments are to address the following questions associated with the use of filtering in the RTI:

- What is the baseline cost (no filtering)?
- What is the best case cost (perfect filtering)?
- What is the host loading caused by the filtering?
- What is the host loading saved by filtering?

These experiments are intended to assess the performance implications of the current RTI implementation, and are intended to provide a basis for the evolution of RTI development.

The type of filtering used in the experiments was data-based filtering. In the data-based filtering approach, each federate indicates to the RTI the classes of objects for which it can provide updates and which classes or attributes it is interested in receiving. Two filter specs were submitted to the RTI for each federate - one for publication and one for subscription. The filter spec defines a filtering scheme which includes:

- a set of extents
- a set of ranges for those extents

The filter space used in the experiment was a two-dimensional geographic filter space.

Since the focus of the experiments was to evaluate filter spaces, it was desirable to use a simulation with deterministic behavior and near-constant system resource requirements. The RTI Analysis Tool (RAT) was constructed for this purpose. The RAT essentially plays back an exercise event log in real-time. During a typical execution, the RAT consumes 65% of the CPU on a Sun Ultra 2.

The scenarios used by the RAT were constructed in a version of ModSAF which was enhanced to output scenarios which can be input to the Scenario Analysis Tool (SAT). The SAT executes the ModSAF scenario and generates an output file which is read by the RAT.

The RAT communicates with the RTI using the Federation Common Software, which encapsulates the RTI API. The FCS provides an object-oriented framework which automates and simplifies federate operations, such as federation initialization, and translation between simulation and federation data representations. The FCS is described in detail in Briggs and Miller[4]. The FCS source code was

instrumented to permit each federate to generate a measures-of-performance(MOP) database on the local host. This data base contains a variety of statistics, including resource utilization for the RTI and RAT processes, federation level latencies, and multicast group usage. After the federation execution is destroyed, the MOP databases for each federate are collected to a central host for post-processing and analysis.

Another goal of the experiment was to eliminate as much possible variability due to system hardware used. For this reason all federate simulations were run on identically configured Sun Ultra 2 systems.

3.0 HYPOTHESES

Before stating the hypotheses for the experiments, some performance metrics will be defined.

3.1 Performance Metrics.

To compare the performance of the federates in the experiment a few metrics were devised. The reduction in the amount of reflected data passed to a federate was one metric used to examine the effectiveness of filter spaces. This savings, S, for a given federate, f, is defined as

$$S_f = \left(\sum_{i=1}^{f-1} u_i + \sum_{i=f+1}^{n} u_i \right) - r_f$$

where n=the number of federates in the federation ($n \geq 2$), u_i= the number of updates issued to the RTI by federate i, and r_f=the number of reflections received by federate f. Normalizing S_f into the range 0..1 allows one to compare federates:

$$S_{fnorm} = \frac{S_f}{\left(\sum_{i=1}^{f-1} u_i + \sum_{i=f+1}^{n} u_i \right)}$$

For federates not using filter spaces, S_f should be close to zero, because federate f should receive reflections for all of the other federates' updates. Note that if the result is exactly zero, all updates sent were reflected. Because packet drops can occur, this value is likely to be greater than zero. Dropped packets are not taken into consideration by this metric. For federates using filter spaces, S_f should be positive, as filtering is expected to reduce the number of reflections sent to a particular federate.

To evaluate the impact on host resources as the number of entities increase, standard Unix system monitoring utilities (ps, netstat, etc.) were used.

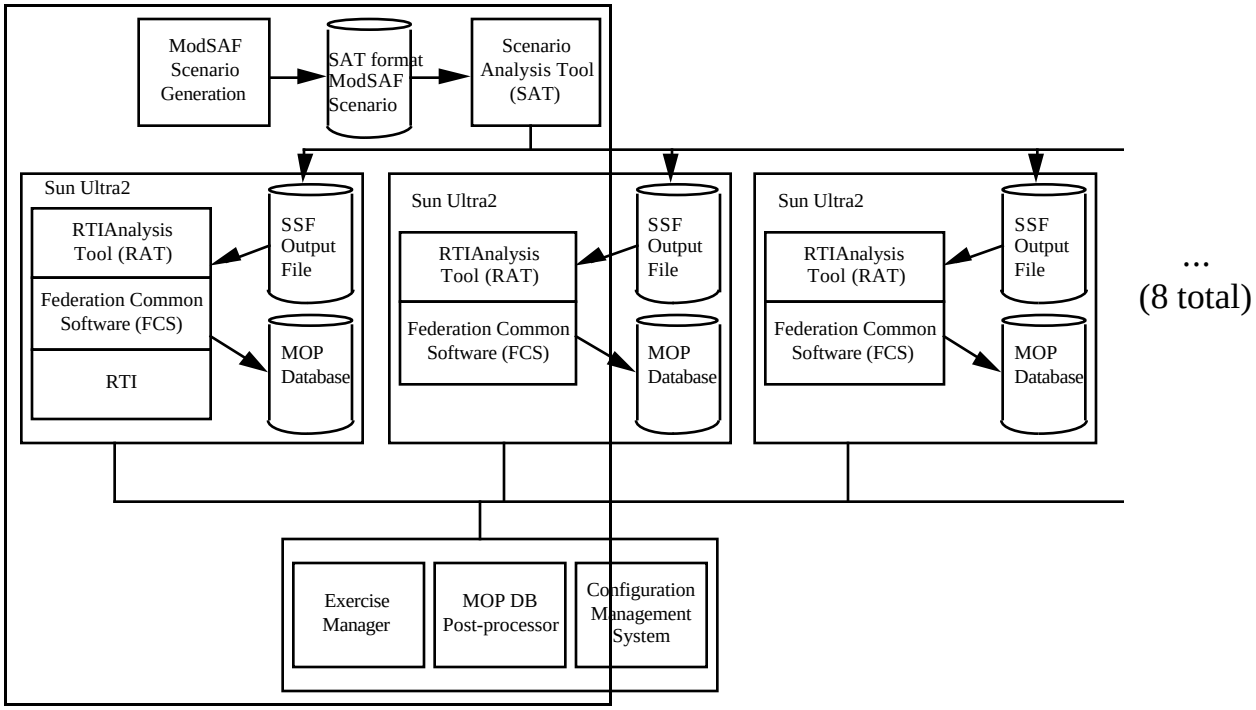


Figure 1. Filter Experiment Configuration.

3.2 Hypothesis 1.
Two-dimensional geographic filtering will reduce the amount of reflections to a federate.

3.3 Hypothesis 2.
Host performance will decrease as the number of simulation entities using filter spaces increases.

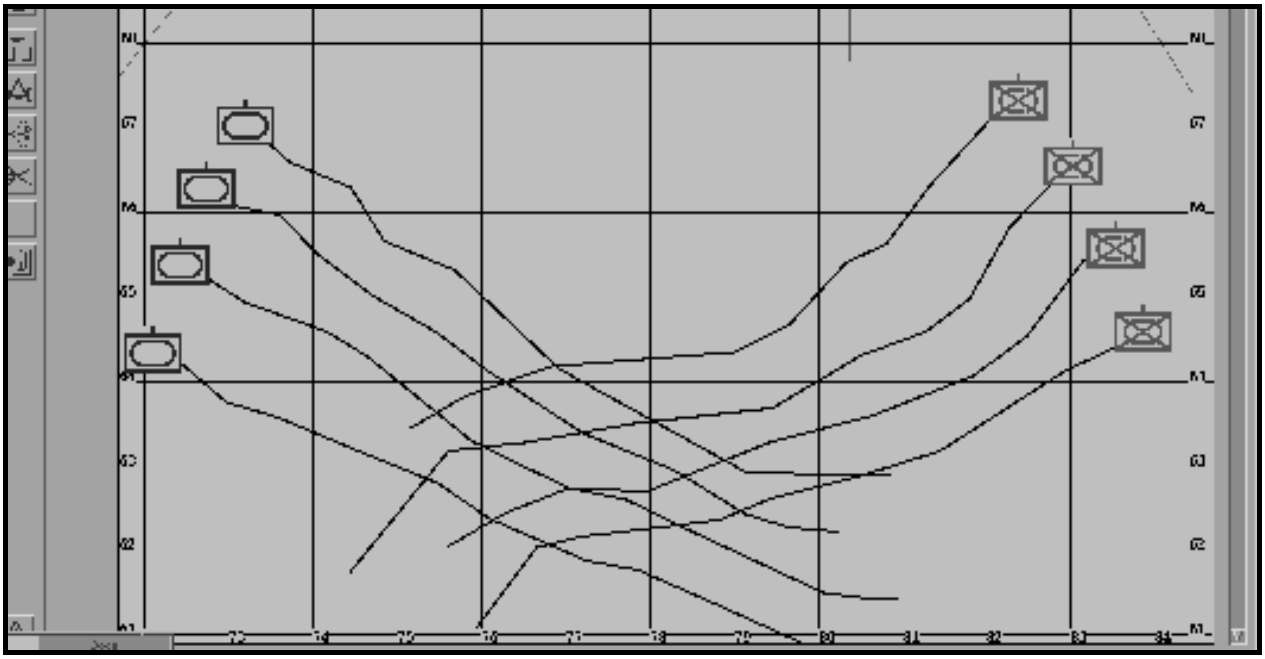


Figure 2. Experiment scenario. Blue units (left) are M1 tank units, red units are T72M tank units (right).

4.0 SYNTHESIS

4.1 Experiment Configuration.

The configuration used in all experiments is depicted in figure 1.

4.1.1 Hardware Configuration.

All software used was run on Sun Ultra 2 systems running Solaris 2.5. Each machine had two 168 MHz UltraSparc processors and 256 MB of RAM. They were networked using a 10-BaseT ethernet. Although an ATM network is available in the HLA Testbed, it was not used as the version 0.33e of the RTI would not work properly with the HLA Testbed's ATM drivers.

4.1.2 Scenario Construction.

The scenario used in all experiments is depicted in Figure 2. In the figure, four red tank units (T-72M's) are advancing from the east on four blue tank units (M-1's). The red and blue forces are initially 10km apart, and their paths begin to cross approximately 13 minutes into the run. The total scenario execution time is approximately 21 minutes.

For all experiment exercises which used filter spaces, a 7 X 7 filter space was defined. Since each grid cell is mapped to a multicast group in the underlying filter space implementation, the total number of multicaast groups possible using the 7 X 7 filter space is 49. This total amount of multicast groups was chosen as a result of some prior experimentation into multicast traffic filtering on Unix workstations. It was found that beyond a certain threshold number of multicast groups, Unix workstations did not perform multicast traffic filtering on the network hardware, but instead would require a kernel interrupt to decide if the host had joined a particular group. In the 0.33e version of the RTI, it was not possible to disable the filter space code Thus for the purposes of comparing executions with and without filtering, the non-filtering runs actually use a 1 X 1 filter space. This "null filter" has the desired effect of allowing all data to pass through the RTI, although with the added processing overhead of executing the filter space code.

The scenarios laydowns and mission assignments were performed in ModSAF, and processed through the SAT synthetic workload tool. The resulting SSF output file is essentially an event log, which is processed by the RAT to provide deterministic stimulation to the RTI. The SSF output file contains entity state updates, interactions (fires and detonates), and communications updates (signal and emission data). For the experiments described below, only the entity states update data was used by the RAT.

4.1.2 Data Collection.

Instrumentation of the federate in the FCS was implemented with the primary goal being to create a data collection system which would be as non-intrusive as possible. Instead, measures-of-performance data is written to a database contained on the local disk. Storing the database locally eliminates the load on the network which would be caused by the sharing of federate MOP data. Time discrepancies between MOP databases are eliminated by using the HLA Testbed's GPS clock synchronization, which provides μsec accuracy. Additionally, writes to the MOP database are buffered to reduce the frequency of disk accesses in the FCS code. These writes are performed on a separate thread of execution in the FCS, so that normal FCS operation is not impacted. After an exercise, the MOP data is collected on a central host and post-processed.

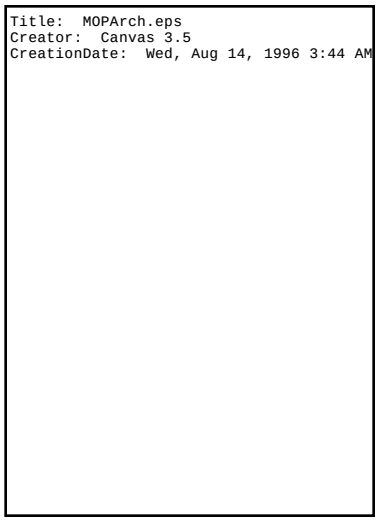


Figure 3. MOP Data Collection Architecture

4.2 Factors and Parameters.

Table 1 lists the factors(things which were varied) in the experiment. Each federate consisted of a single host machine.

Factor	Settings
No. of Hosts (federates)	2, 4, 8
No. of Entities	50, 100, 150
Filter Spaces	Off, On

Table 1. Experiment Factors.

Table 2 summarizes some of the important parameters in the experiment.

Parameter	Setting
Hardware Platform	Sun UltraSparc 2
Network Hardware	10-base T Ethernet
RTI Version	0.33e

Orbix Version	1.3
Federate Simulations	RAT
RAT run mode	realtime, no interactions
Filter space	2D Geographic
Sensor Range	1.5km
Playbox	10km X 10km
Entity Types	M1(blue forces), T72M (red forces)

Table 2. Experiment Parameters.

4.3 Experiments Performed.

4.3.1 Experiment 1 - Federate/Host Scaling

The purpose of this experiment was to evaluate hypothesis 1 by examining filter space behavior with a varying number of hosts. The entities were divided among the hosts as listed in table 3 by equally dividing the units.

The complete set of data for the eight federate runs was not available at the time of this writing.

4.3.2 Experiment 2 - Entity Scaling

The purpose of this experiment was to evaluate hypothesis 2. The desired total number of entities was achieved by varying the number of entities per aggregate unit, rather than by adding units. The number of entities on each force was equal in all cases (24, 48, 74).

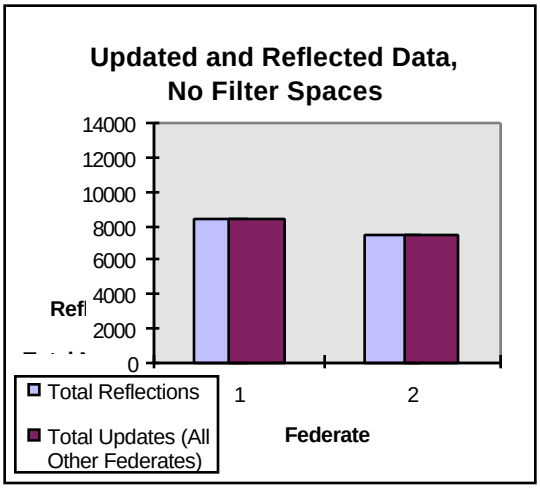
5.0 VALIDATION

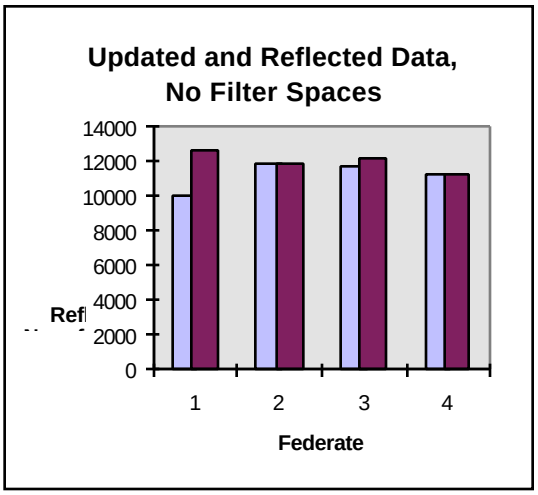
5.1 Federate/Host Scaling Results

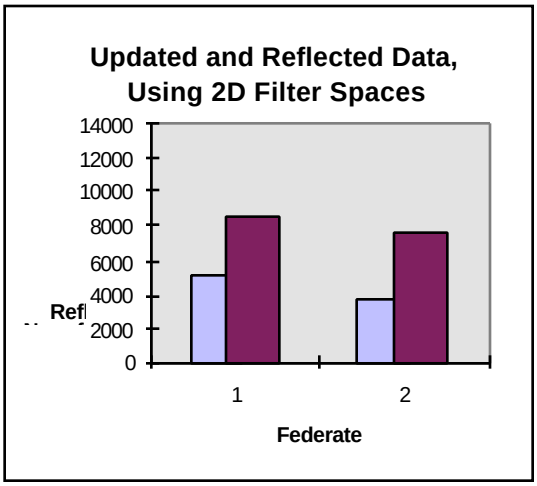
Figure 3 plots the total number of reflections and updates for each federate in an exercise. Results are shown for a 100 entity, two federate exercise and a 100 entity, four federate exercise. In each plot, a "total reflections" bar indicates the total number of reflection messages which were received by that federate. For a given federate, the "total updates" bar indicates the total number of updates issued to the RTI by all *other* federates. For example,. the total number of updates for federate 2 is equal to the sum of the total number of updates for federates 1, 3, and 4. As described in section 3.1, these difference between reflections and total updates should be close to zero when no filtering is performed.

The plots on the left show the behavior of the RTI with no filtering (this is given as a baseline), while the plots on the right show the effects of filter spaces. For the plots on the right, S_1=.389 and S_2=0.495 in the two federate exercise, while S_1=.599, S_2=.290, S_3=.310 and S_4=.480 for the four federate exercise. In all cases, use of filter spaces caused a reduction in the number of reflections to the federate.

In the four federate exercise, it is clear that a significant number of packets are dropped for federate one, in both the filter space and non-filter space runs. In all experiments federate one was running the RTI executive *in addition to* the RAT and local RTI clients.







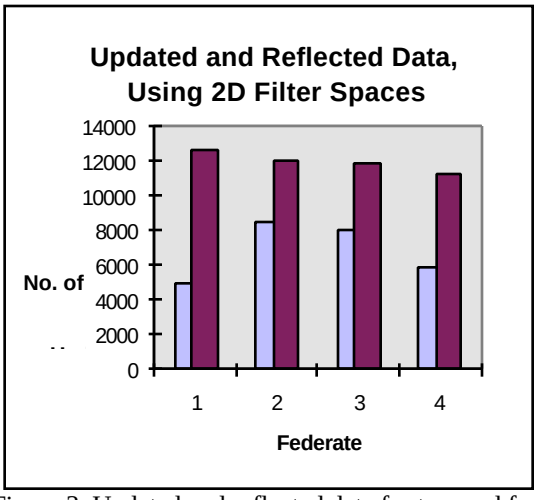


Figure 3. Updated and reflected data for two and four federate exercises.

5.2 Entity Scaling Results.

In this experiment, the configuration of Figure 1 was tested using a varying number of simulation entities. As mentioned above, this test was performed to get a sense of the trends in RTI resource utilization as the number of entities is increased.

The results from these tests were not as conclusive as those for experiment 1. During runs of the 150 entity tests, the system would lock up early on in the exercise, which prohibited more extensive testing. Some sense of the trends in CPU and memory usage can be seen in figures 4 through 6. The most notable trend is the rapid increase in memory consumption by the dm_local_server process, which is presumed to have a memory leak in 0.33e of the RTI. This problem is underscored by Figure 7, which plots memory usage vs. time for each RTI process and the RAT for a single federate in the 100 entity exercise. In this plot the memory usage samples were taken at five second intervals, and thus the value on the x axis must be multiplied by five to get the exercise time. Similar behavior was observed for other federates.

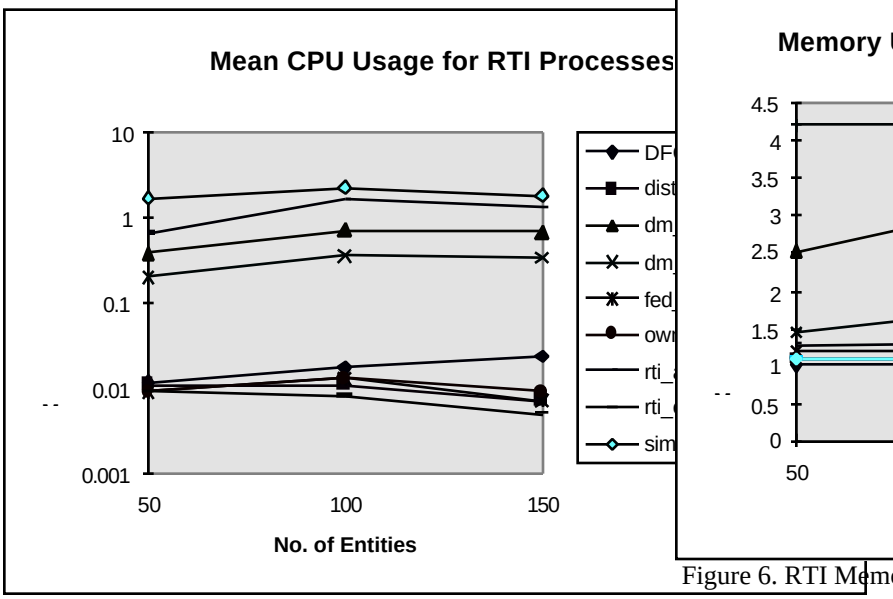


Figure 4. RTI CPU Usage by process.

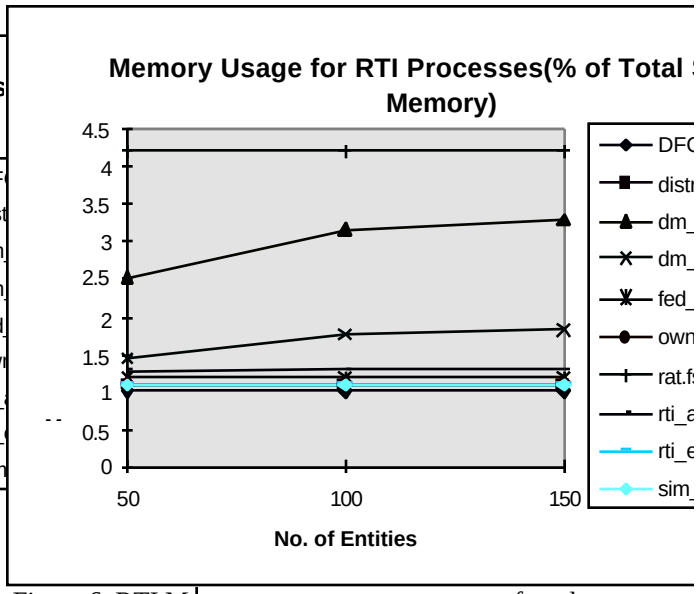


Figure 6. RTI Memory usage as a percentage of total system memory.

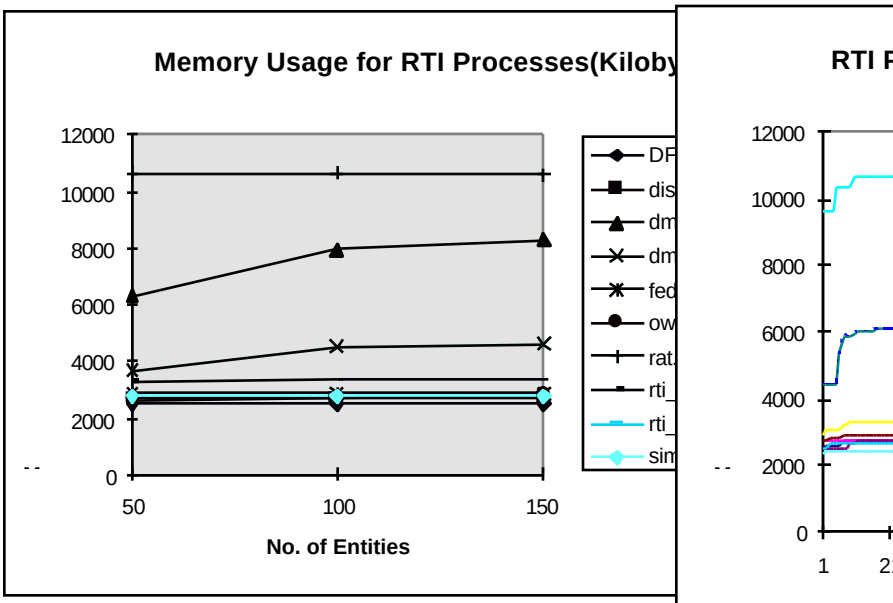


Figure 5. RTI Memory Usage.

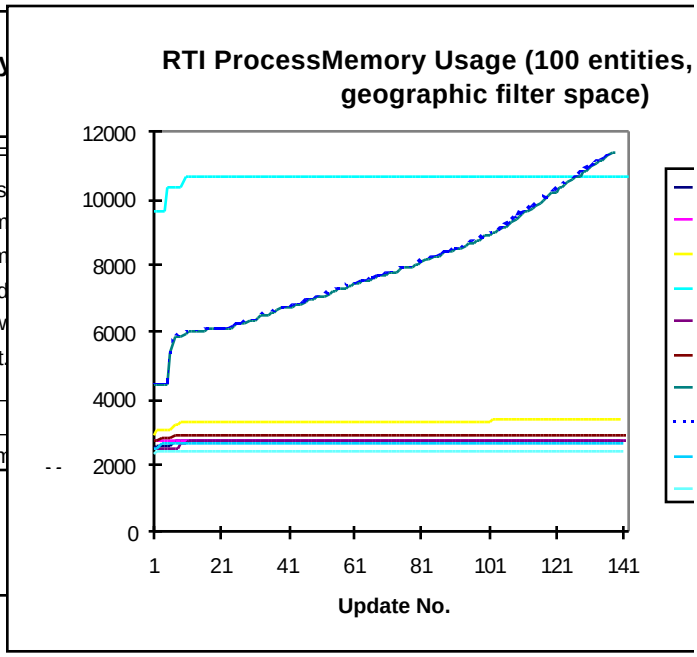


Figure 7. RTI Process Memory Usage.

6.0 CONCLUSIONS

The results from the study indicate that the use of filter spaces does reduce the amount of reflected data in a system. Although, the results are derived from a single scenario, they do show promise as a way of reducing the amount of network traffic required for a large distributed exercise. Further investigations will address more complex scenarios which include fast moving entities and wide-area viewers(WAVs).

The results from the study into the impact of filtering on system resource utilization are not as conclusive, and further experimentation and data analysis in this area is necessary.

Other areas of interest include examining other filtering schemes (source based, destination-based), examining the performance of using actual simulations (e.g. ModSAF), examining the behavior of the RTI over longer scenarios, and examining the performance on other host architectures. Some of this work has been performed or is underway in the HLA testbed, including experimentation with an HLA integration of ModSAF and the RTI.

7.0 Acknowledgments

The authors would like to thank Dave Meyer, Mike Mazurek, Gordon Miller, Jeff Pace, Russ Richardson, Dan Van Hook and Sudhir Srinivasan for their contributions to this project.

8.0 REFERENCES

[1] Defense Modeling and Simulation Office, "Department of Defense High Level Architecture Interface Specification", Version 0.4, 7 March, 1996.

[2] Van Hook, D., "Filter Spaces for RTI Declaration Management". White paper ASD-15-084, 15th Workshop on Standards for Interoperability of Defense Simulations, Orlando FL, September 1996.

[3] Van Hook, D. and Rak, Steven, "Evaluation of Grid-Based Relevance Filtering for Multicast Group Assignment". White paper ASD-96-106, 14th Workshop on Standards for Interoperability of Defense Simulations, Orlando FL, March 1996.

[4] Briggs, R. and Miller, G., "The JPSD Experiment Common Software", White paper ASD-15-095, 15th Workshop on Standards for Interoperability of Defense Simulations, Orlando FL, September 1996.

[5] Mellon, Larry F. "Intelligent Addressing and Routing of Data Via the RTI Filter Constructs", White Paper ASD-96-096, 15th Workshop on Standards for Interoperability of Defense Simulations, Orlando FL, September 1996.

9.0 ABOUT THE AUTHORS

Jeff Olszewski is a senior computer scientist in the Simulation Technology Division of SAIC, and is currently working in support of the Synthetic Theater of War (STOW) program. He received his B.Sc. degree in computer science from the University of Pittsburgh, and will complete his M. Sc. in Computer Science from George Washington University in 1997.

Richard A. Briggs is a Systems Engineer with Virtual Technology Corporation. He received his B.Sc. in Computer Science from the Pennsylvania State University. He is currently the VTC lead for the RTI Integrated Product Team which is tasked to productize the RTI for the 1.0 release. He was the lead engineer for the JPSD Experiment and the Federation Common Software.

Larry Mellon is a senior computer scientist and branch manager with Science Applications International Corporation(SAIC). He received his B.Sc. degree from the University of Calgary. His research interest include parallel simulation and distibuted systems. He is a lead architect for the ARPA funded Synthetic Theater of War Program.